Please amend the Specification as follows:

Please amend the present title as follows:

Two-~~stage~~-phase commit with queryable caches

Please amend page 6, lines 6-8 as follows:

a1

The two-~~stage~~-phase commit process used for transactions between distributed databases needs to be optimized for the situation where the transaction does not result in modifications to particular ones of the distributed databases.

Please amend page 6, lines 15-24 as follows:

a2

The problem of optimizing the two-~~stage~~-phase commit protocol is solved as follows: in general, when an action is carried out in a distributed system, a component of the distributed system that is involved in the action is a coordinator for the action and other components involved in the action are cohorts for the action. During the action, the cohorts send messages which are available to the coordinator. In the optimization, each cohort augments messages which are available to the coordinator with information which indicates relevant state of the cohort with regard to the action. The coordinator reads the messages and retains the most recent relevant state for each cohort and performs an action according to the relevant state.

Please amend page 6, line30-page 7 line 5 as follows:

a3

When the protocol is a two-~~stage~~-phase commit protocol, the action performed by the coordinator is sending an *abort* message according to the two-stage commit protocol to a cohort when the retained state for the cohort indicates that the transaction will not modify data in the cohort.

Please amend page 16, lines 5-21 as follows:

a4

In the database context, the two-phase commit protocol works like this: one of the distributed databases is the *coordinator* for the database transaction for which the protocol is being used; the other distributed databases are the *cohorts*. It should be noted here that coordinator and cohort need not be fixed roles. In many cases, the database at which a transaction is initiated

will be the coordinator for that transaction and the rest of the databases will be the cohorts for the transaction. The coordinator and all of the cohorts have *redo* logs; whenever the database system receives a statement which modifies the database, it is written to the database system's redo log. The statement itself is executed on a temporary copy of the database object involved. Additionally, each database system has a commit log in which it records the state of the two-~~stage~~ phase commit for the transaction. Phase one begins when something occurs in the coordinator which marks the end of the transaction. In response to the end of the transaction, the coordinator sends a *commit request* message for the transaction. Each cohort then reads the redo log for the transaction and determines whether it can perform the transaction. If the cohort can, it sends an *agree* message to the coordinator; if the cohort cannot perform the transaction, it sends an *abort* message to the coordinator. Each of the database systems also marks its commit log to indicate the state of the commit operation.

Please amend page 17, lines 27-30 as follows:

In systems with database caches such as system 201, most of the transactions on cache 203 are read operations. Thus, substantial savings in system resources are possible if the coordinator knows at the end of a transaction whether it is necessary at all to perform the two-~~stage~~ phase commit protocol.

Please amend page 19, lines 12-24 as follows:

FIG. 5 shows how a preferred embodiment of the invention operates in the coordinator; FIG. 6 shows how it operates in a cohort. It should be noted here that the only difference between what the cohort does in the unoptimized protocol and what it does in the optimized protocol is that it augments the messages it sends to the application program with the read-only status information 407. Flowchart ~~503~~ 501 begins with the start of a new transaction (503); at 505, the coordinator sets transaction status information 413 to read-only. Then the coordinator executes loop 513 until an event occurs which terminates the transaction (507). In loop 513, the coordinator receives each cohort message 403 in turn (515); on receiving the message, it writes current cohort status info 405 to current cohort status 413 in the outgoing link-transaction object for the cohort and transaction. Current cohort status 413 for a link-transaction object thus always contains the most recent read-only state of the cohort.

Please amend page 20, line 28-page 21, line 4 as follows:

As can be seen from the foregoing, the optimization replaces the messages of the two-phase commit protocol with the single abort message when a cohort and any subtree of which it is the root is read only; moreover, the optimization eliminates for read-only subtrees the bookkeeping which the coordinator and local coordinators would otherwise do for them in the course of the commit protocol and further eliminates the bookkeeping for the two-phase commit protocol done in read-only cohorts by the cohorts themselves. When the cohorts have high percentages of read-only data, as is typical of caches, the two-~~stage~~ phase commit protocol can often be reduced to an abort message sent to the cohort.

Please amend page 21, lines 8-15 as follows:

The use of the foregoing technique for optimizing two-~~stage~~ phase commit with operations that do not change the state of the cohort is not limited to situations where the coordinator is a database cache and the cohort is a database from which the objects in the cache have been copied. For example, a cache such as database 203 is a member of a distributed database system that employs replication; consequently, if a transaction is performed on replicated data in another of the database systems in the distributed database system, the database system which first receives the transaction is the coordinator and the other database systems in the distributed system, including database 203, are cohorts.

Please amend page 22, lines 10-13 as follows:

~~/** ideas for other optimizations? **/~~

~~Other applications of the optimization include~~

~~/** ideas for other applications? **/~~

Please amend the claims as follows:

Please cancel claim 1.